UNIVERSITÄT
LEIPZIG

Introduction to Combinatorics

Dat Tran (FMI, Leipzig University)

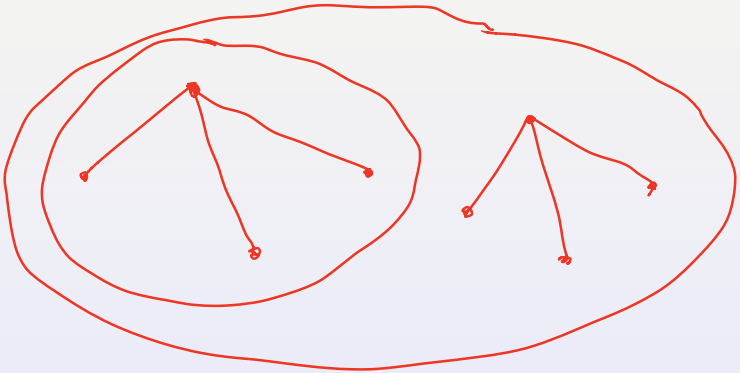
SoSe 2020 (Day 8 - 26/05/2020)

Contents

Definition

A connected graph G is a tree if it is acyclic, that is, it has no cycles. An acyclic graph is called a forest.

Example



Theorem

Every tree T is bipartite.

Proof.

□

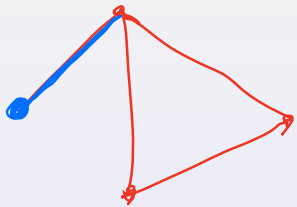
We use the Thm: G is bipartite $\Leftrightarrow$ all cycle has even length.

T has no cycle $\Rightarrow$ T is bipartite.

Definition

A vertex of degree one is called a pendant vertex, and the edge incident to it is a pendant edge.

Example



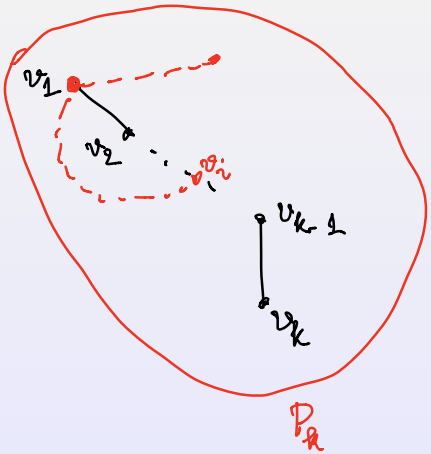
Theorem

Every tree on $n \geq 2$ vertices has at least one pendant vertex.

Proof.

□

$P_k = \{v_1, \dots, v_k\}$ is a longest path in T



$d(v_1) = 1$.

v_1 : pendant vertex.

Theorem

A tree on n vertices has exactly n − 1 edges.

Proof.

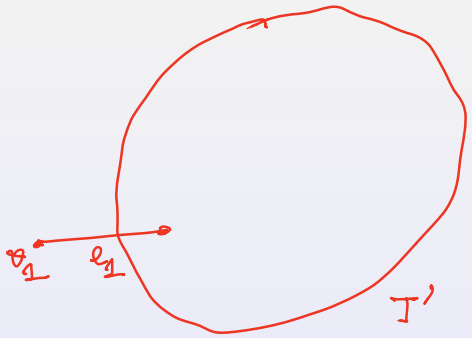
□

By induction on n :

* $n = 1$: $T_1 =$ • ✓

Ind. ass.: * T_{n-1} has exactly $n-2$ edges

T_n is a tree $\Rightarrow \exists\, v_1$ pendant vertex



delete $\{v_1, e_1\}$ from $T_n \Rightarrow T'$ a tree on $n-1$ vertices

$\Rightarrow T'$ has exactly $n-2$ edges.

$\Rightarrow T_n$ has exactly $n-1$ edges.

Theorem

A tree with a vertex of degree $k \geq 1$ has at least k pendant vertices. In particular, every tree on at least two vertices has at least two pendant vertices.

Proof.

□

Claim 1. T : tree on n vertices. $V(T) = \{v_1, \dots, v_n\}$

$v_j \in T$: $d(v_j) = k \geq 1 \overset{?}{\Longrightarrow}$ # {pendant vertices} $\geq k$.

- $k = 1$: ✓
- $k \geq 2$: $I = \{i \in [n] : d(v_i) = 1\} \not\ni j$

$$\sum_{i=1}^{n} d(v_i) = 2|E| = 2(n-1)$$

$$k + |I| + \sum_{i \notin I \cup \{j\}} d_i \geq k + |I| + 2(n - |I| - 1)$$

($d_i \geq 2$)

Claim 2. T_2 : •—•

T_n $(n \geq 3)$

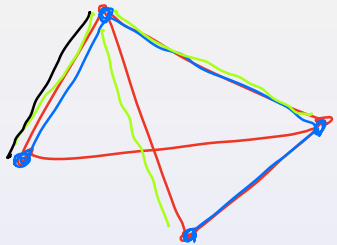
$\exists v \in T$: $d(v) \geq 2 \Rightarrow$ # pendant vertices ≥ 2 (Claim 1)

$\Rightarrow |I| \geq k$.

Definition

If G is a connected graph on n vertices, a spanning tree for G is a subgraph of G that is a tree on n vertices.

Example



Theorem

Every connected graph has a spanning tree.

Proof.

□

Remark

In general, spanning trees are not unique, that is, a graph may have many spanning trees. Some edges could be in every spanning tree whenever there are multiple spanning trees. For example, any pendant edge must be in every spanning tree, as must any edge whose removal disconnects the graph (such an edge is called a bridge.)

Theorem

If G is connected, it has at least $n-1$ edges; moreover, it has exactly $n-1$ edges if and only if it is a tree.

Proof.

$\square$

G connected $\Rightarrow |E(G)| \geq n-1$

"$=$" $\Leftrightarrow$ G is a tree.

$\exists T$: spanning tree $\Rightarrow |E(T)| = n-1$

$= |V(T)| = n$

$E(T) \subseteq E(G)$

$\Rightarrow |E(G)| \geq n-1$.

($\Leftarrow$) G is a tree $\Rightarrow |E(G)| = n-1$

($\Rightarrow$) $|E(G)| = n-1 \Rightarrow G \equiv T$

$\square$

Theorem

G is a tree if and only if there is a unique path between any two vertices.

Proof.

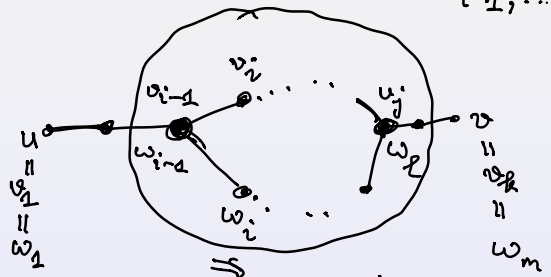
□

($\Rightarrow$) G is a tree $\Rightarrow$ G is connected $\Rightarrow$ $\forall u, v \in G$ $\exists$ a path P_k connect u and v

$P_k = \{v_1, \dots, v_k\}$

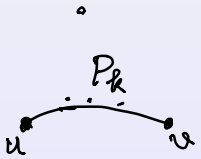
Assume that $\exists$ P'_s connect u and v.

$\{w_1, \dots, w_m\}$



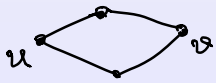
$\Rightarrow$ contradiction to G is a tree $\Rightarrow \nexists$!

($\Leftarrow$)



$\forall u, v \in G$ $\exists$ path connect u and v $\Rightarrow$ G is connected.

Assume that $\exists$ a cycle in G

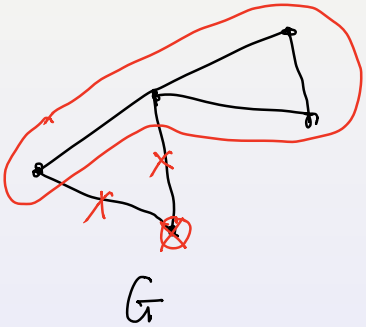


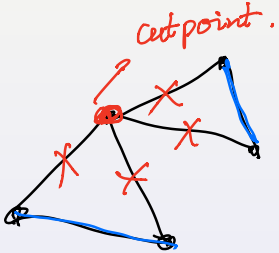
$\Rightarrow \geq 2$ paths connect u and v $\lightning$.

Definition

A cutpoint in a connected graph G is a vertex whose removal disconnects the graph.

Example



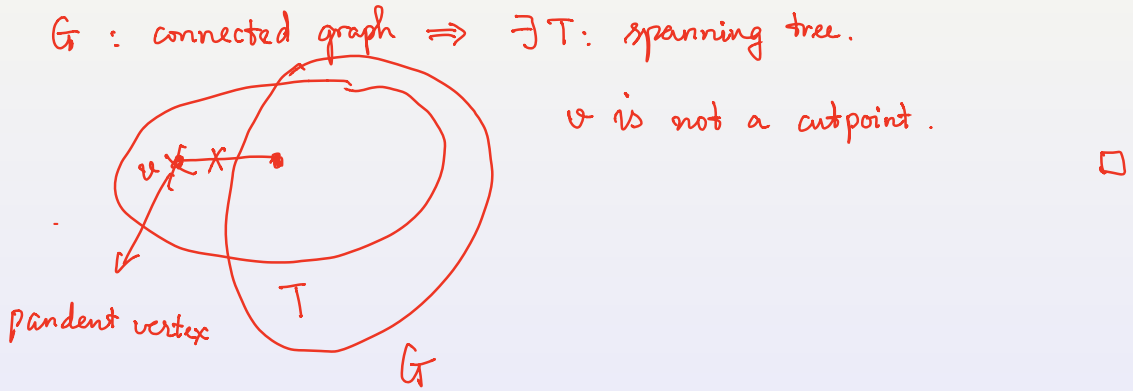


Theorem

Every connected graph has a vertex that is not a cutpoint.

Proof.

□

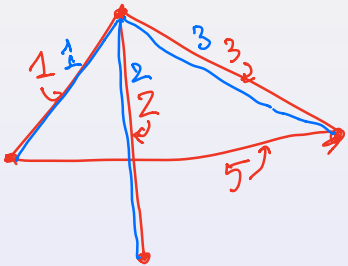


In some applications, a graph G is augmented by associating a weight or cost with each edge; such a graph is called a weighted graph. For example, if a graph represents a network of roads, the weight of an edge might be the length of the road between its two endpoints, or the amount of time required to travel from one endpoint to the other, or the cost to bury cable along the road from one end to the other. In such cases, instead of being interested in just any spanning tree, we may be interested in a least cost spanning tree, that is, a spanning tree such that the sum of the costs of the edges of the tree is as small as possible. For example, this would be the least expensive way to connect a set of towns by a communication network, burying the cable in such a way as to minimize the total cost of laying the cable.

Definition

A weighted graph is a graph G together with a cost function $c : E(G) \to \mathbb{R}_{>0}$.
If H is a subgraph of G, the cost of H is $c(H) = \sum_{e \in E(H)} c(e)$.

Example



G

$c : E(G) \to \mathbb{R}_{>0}$

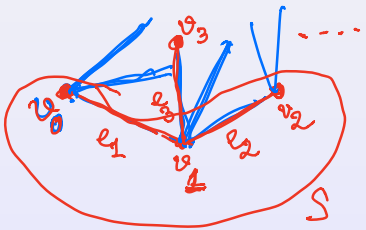
H : subgraph

$c(H) = 1 + 2 + 3$

The Jarník Algorithm:

Given a weighted connected graph G, we construct a minimum cost spanning tree T as follows:

- Choose any vertex $v_0 \in G$ and include it in T.
- If vertices $S = \{v_0, v_1, \ldots, v_k\}$ have been chosen, choose an edge with one endpoint in S and one endpoint not in S and with smallest weight among all such edges. Let v_{k+1} be the endpoint of this edge not in S, and add it and the associated edge to T.
- Continue until all vertices of G are in T.



$e_{n-1}, v_{n-1} \Rightarrow T = (\{v_0, \ldots, v_{n-1}\}, \{e_1, \ldots, e_{n-1}\})$

Theorem

The Jarník Algorithm produces a minimum cost spanning tree.

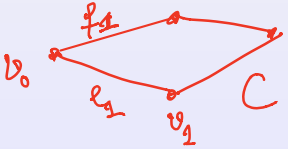
Proof.

$T = (\{v_0, \dots, v_{n-1}\}, \{e_1, \dots, e_{n-1}\})$: spanning tree after the algorithm.

Assume that T_{min} : optimal spanning tree. We would like to prove $c(T_{min}) = c(T)$.

We construct a sequence of spanning trees $T_0 = T_{min}, T_1, \dots, T_{n-1} = T$.

* $e_1 \in T_{min} = T_0$, set $T_1 = T_0$
* $e_1 \notin T_0$, we add e_1 to T_0 $\Rightarrow$ $T_0 \cup \{e_1\}$ has a cycle $C \ni e_1$



$\Rightarrow \exists f_1 \in C, \begin{cases} f_1 \sim v_0 \\ f_1 \neq e_1 \end{cases}$

$c(e_1) \leq c(f_1)$

$T_1 = T_0 \cup \{e_1\} \setminus \{f_1\}$

$\Rightarrow$ spanning tree

* $c(e_1) < c(f_1) \Rightarrow c(T_1) < c(T_0)$ >< optimal spanning tree of T_0

$\Rightarrow c(e_1) = c(f_1) \Rightarrow c(T_1) = c(T_0)$.

$T_0 \longrightarrow T_1 = T_0$

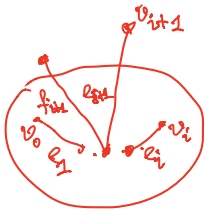
$\searrow T_1 = T_0 \cup \{e_1\} - \{f_1\}$

...

T_i : $c(T_i) = c(T_0) = \dots = c(T_{i-1})$

* $e_{i+1} \in T_i \Rightarrow T_{i+1} = T_i$

* $e_{i+1} \notin T_i$



S_i

$\Rightarrow \{e_{i+1}\} \cup T_i$ has a cycle $C \ni e_{i+1}$

$\Rightarrow \exists f_{i+1} \in C : f_{i+1} \neq e_{i+1}$

$T_{i+1} = T_i \cup \{e_{i+1}\} - \{f_{i+1}\}$

$c(T_{i+1}) = c(T_i) \overset{\text{induction}}{\Rightarrow} c(T_{n-1}) = c(T_0)$

$c(T) \qquad c(T_{min})$

□.